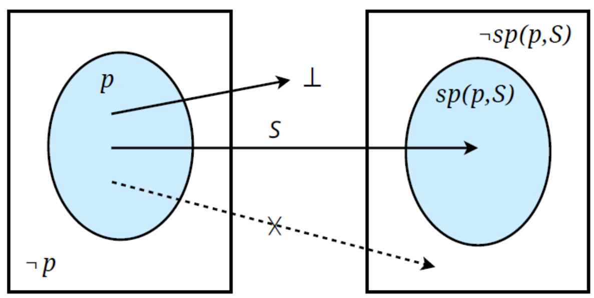
The Strongest Postcondition

- Given a precondition p and program S, **the strongest postcondition** of p and S , written as $sp(p, S)$ is (the predicate that stands for) the set of states we could terminate in if we run S starting in a state that satisfies p.
  - In symbols, using the language of states: $sp(p, S) = \{\tau \mid \tau \in M(S, \sigma) - \bot \text{ for some } \sigma \text{ where } \sigma \vDash p\}$, or equivalently $sp(p, S) = \ \cup_\sigma (M(S, \sigma) - \bot) \text{ where } \sigma \vDash p$.

  - From the definition of the strongest postcondition we can see that $\vDash \{p\}\ S\ \{sp(p, S)\}$; in other words, this postcondition only guarantees a valid triple under partial correctness. The reason is easy to see: the precondition p is given, and there could be states satisfying p make program S diverge or create error.



- From the above figure, we can see:
  - If $\sigma \vDash p$, then $\text{for all } \tau \in M(S, \sigma), \text{either } \tau = \bot \text{ or } \tau \vDash sp(p, S)$.
  - If $\sigma \nvDash p$, we don't know anything interesting about $M(S, \sigma)$ and $sp(p, S)$.

1. Prove that $\vDash \{p\}\ S\ \{q\}\ \textbf{iff}\ sp(p, S) \Rightarrow q$.
   It looks trivial, but our definition of $sp(p, S)$ didn't say anything about this postcondition the strongest. In this example, let us prove that $sp(p, S)$ is the strongest postcondition.

   - $\Leftarrow$: By the definition of $sp(p, S)$, we have $\vDash \{p\}\ S\ \{sp(p, S)\}$. And since $sp(p, S) \Rightarrow q$, then $\vDash \{p\}\ S\ \{q\}$ (weakening the postcondition).

   - $\Rightarrow$: Let τ be a state such that $\tau \vDash sp(p, S)$. By the definition of $sp(p, S)$, there exists some $\sigma \vDash p$ such that $\tau \in M(S, \sigma) - \bot$. Since $\vDash \{p\}\ S\ \{q\}$ implies that $M(S, \sigma) - \bot \vDash q$, thus $\tau \vDash q$. To sum up, we get "if $\tau \vDash sp(p, S)$, then $\tau \vDash q$", which implies "$sp(p, S) \Rightarrow q$".

Calculate *sp* for Loop-free Programs

Like wlp, we can use some algorithm/rules to calculate $sp(p, S)$ textually.

- $sp(p, \textbf{skip}) \equiv p$.
- $sp(p, v \coloneqq e) \equiv p[v_0 / v] \wedge v = e[v_0 / v]$, where v_0 is the aged v (in other words, the old value of v before executing $v \coloneqq e$).
  - This is the forward assignment rule, so actually this rule can produce the strongest postcondition.

- $sp(p, S_1; S_2) \equiv sp(sp(p, S_1), S_2)$.

2. Calculate the following sp's.
   a. $sp\ (x > y,\ x \coloneqq x + k) \equiv (x > y)[x_0\ /\ x] \wedge x = (x + k)[x_0\ /\ x] \equiv x_0 > y \wedge x = x_0 + k$
   b. $sp\ (x_0 > y \wedge x = x_0 + k,\ y \coloneqq y + k) \equiv x_0 > y_0 \wedge x = x_0 + k \wedge y = y_0 + k$

   c. $sp\ (x > y,\ x \coloneqq x + k;\ y \coloneqq y + k) \equiv x_0 > y_0 \wedge x = x_0 + k \wedge y = y_0 + k$ #Combine a. and b.
   - By losing x_0 and y_0, we can slightly weaken the postcondition to $x > y$.

   d. $sp(x > f(x, y), x \coloneqq x + 1; x \coloneqq x + x)$
   - $sp(x > f(x, y),\ x \coloneqq x + 1)$ $\equiv \big(x > f(x, y)\big)[x_0\ /\ x] \wedge x = (x + 1)[x_0\ /\ x]$
     $\equiv x_0 > f(x_0, y) \wedge x = x_0 + 1$

   - $sp(x > f(x, y), x \coloneqq x + 1; x \coloneqq x + x)$
     $\equiv sp(x_0 > f(x_0, y) \wedge x = x_0 + 1,\ x \coloneqq x + x)$
     $\equiv (x_0 > f(x_0, y) \wedge x = x_0 + 1)[x_1\ /\ x] \wedge x = (x + x)[x_1\ /\ x]$
     $\equiv x_0 > f(x_0, y) \wedge x_1 = x_0 + 1 \wedge x = x_1 + x_1$

- Let us think about sp in a conditional statement with an example:
  $sp(T,\ \textbf{if } x \geq y + z \textbf{ then } x \coloneqq x - 1 \textbf{ else } y \coloneqq y + 2 \textbf{ fi}) \equiv ?$

  Following intuition, it is quite straightforward to come up with the following solution:
  - When the if condition is true, we should have $sp(T \wedge x \geq y + z,\ x \coloneqq x - 1) \equiv T \wedge x_0 \geq y + z \wedge x = x_0 - 1$.
  - When the if condition is false, we should have $sp(T \wedge x < y + z,\ y \coloneqq y + 2) \equiv T \wedge x < y_0 + z \wedge y = y_0 + 2$.
  - The sp for the whole statement should one of the above, thus:
    $"sp"(T,\ \textbf{if } x \geq y + z \textbf{ then } x \coloneqq x - 1 \textbf{ else } y \coloneqq y + 2 \textbf{ fi})$
    $\equiv (T \wedge x_0 \geq y + z \wedge x = x_0 - 1) \vee (T \wedge x < y_0 + z \wedge y = y_0 + 2)$

  - Is this postcondition the strongest? No, it can be stronger since we didn't include that y is not updated in the true branch and x is not updated in the false branch. We can add this information by aging more variables.

- To calculate the sp for a conditional statement, we need to calculate some variable sets first:
  - $lhs(S) =$ the set of variables that appear as the lhs of assignments in statement S.
  - $rhs(S) =$ the set of variables that appear as the rhs of assignments in statement S.
  - $free(p) =$ the set of variables that are free in precondition p.
  - $aged(p, S) = lhs(S) \cap \big(rhs(S) \cup\ free(p)\big)$ is the set of variables whose assignments cause aging.

- Let $IF \equiv\ \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}$, and let $aged(p, IF) = \{x, y, \dots\}$. Then $sp(p, IF) \equiv sp(p_0 \wedge B,\ S_1) \vee sp(p_0 \wedge \neg B,\ S_2)$, where $p_0 = p \wedge x = x_0 \wedge y = y_0 \dots$

- Let $NF \equiv\ \textbf{if } B_1 \rightarrow S_1\ \ \square\ B_2 \rightarrow S_2 \textbf{ fi}$, and let $aged(p, NF) = \{x, y, \dots\}$. Then $sp(p, NF) \equiv sp(p_0 \wedge B_1,\ S_1) \vee sp(p_0 \wedge B_2,\ S_2)$, where $p_0 = p \wedge x = x_0 \wedge y = y_0 \dots$

3. Calculate $sp(T,\ \textbf{if } x \geq y + z \textbf{ then } x \coloneqq x - 1 \textbf{ else } y \coloneqq y + 2 \textbf{ fi})$
   - Let $p \equiv T, S \equiv\ \textbf{if } x \geq y + z \textbf{ then } x \coloneqq x - 1 \textbf{ else } y \coloneqq y + 2 \textbf{ fi}$
     - $lhs(S) = \{x, y\}$
     - $rhs(S) = \{x, y\}$

    - $free(p) = \emptyset$
    - $aged(p, S) = \{x, y\}$

  - $sp(T \land x = x_0 \land y = y_0 \land x \geq y + z,\ x \coloneqq x - 1)$
    $\equiv T \land x_0 = x_0 \land y = y_0 \land x_0 \geq y + z \land x = x_0 - 1$

  - $sp(T \land x = x_0 \land y = y_0 \land x < y + z,\ y \coloneqq y + 2)$
    $\equiv T \land x = x_0 \land y_0 = y_0 \land x < y_0 + z \land y = y_0 + 2$

  - $sp(p, S)$
    $\equiv (T \land x_0 = x_0 \land y = y_0 \land x_0 \geq y + z \land x = x_0 - 1) \lor (T \land x = x_0 \land y_0 = y_0 \land x < y_0 + z \land y = y_0 + 2)$
    $\Leftrightarrow (y = y_0 \land x_0 \geq y + z \land x = x_0 - 1) \lor (x = x_0 \land x < y_0 + z \land y = y_0 + 2)$

4. Calculate $sp(p, S)$ where $p \equiv (x = y)$ and $S \equiv \mathbf{if}\ y \geq 1 \rightarrow x \coloneqq 1\ \square\ y \leq 1 \rightarrow z \coloneqq 0\ \mathbf{fi}$.
   - $lhs(S) \equiv \{x, z\}$
   - $rhs(S) \cup free(p) \equiv \{x, y\}$
   - $aged(p, S) \equiv \{x\}$

   - $sp(x = y \land x = x_0 \land y \geq 1,\ x \coloneqq 1) \equiv x_0 = y \land x_0 = x_0 \land y \geq 1 \land x = 1$
   - $sp(x = y \land x = x_0 \land y \leq 1,\ z \coloneqq 0) \equiv x = y \land x = x_0 \land y \leq 1 \land z = 0$
   - $sp(p, S) \equiv (x_0 = y \land x_0 = x_0 \land y \geq 1 \land x = 1) \lor (x = y \land x = x_0 \land y \leq 1 \land z = 0)$

<u>Forward Assignment vs. Backward Assignment</u>

- With backward assignment rule, we can get partially valid triple $\{q[e \mathbin{/} v]\}\ v \coloneqq e\ \{q\}$; and with forward assignment rule we get partially valid triple $\{p\}\ v \coloneqq e\ \{p[v_0 \mathbin{/} v] \land v = e[v_0 \mathbin{/} v]\}$. What if we apply the "opposite" assignment rules on each of these two triples?

- First, let us calculate the $sp(q[e \mathbin{/} v], v \coloneqq e)$, where this precondition is calculated from the backward assignment rule.
  - $sp(q[e \mathbin{/} v], v \coloneqq e) \quad \equiv q[e \mathbin{/} v][v_0 \mathbin{/} v] \land v = e[v_0 \mathbin{/} v]$
    $\Leftrightarrow q[e[v_0 \mathbin{/} v] \mathbin{/} v] \land v = e[v_0 \mathbin{/} v]$
    $\Rightarrow q[v \mathbin{/} v]$
    $\Leftrightarrow q$
    - $sp(q[e \mathbin{/} v], v \coloneqq e) \Rightarrow q$. This implies that $\{q[e \mathbin{/} v]\}\ v \coloneqq e\ \{q\}$ is a valid triple under partial correctness. Note that, we don't have $sp(q[e \mathbin{/} v], v \coloneqq e) \Leftrightarrow q$.

- Then, let us calculate the $wlp(v \coloneqq e,\ p[v_0 \mathbin{/} v] \land v = e[v_0 \mathbin{/} v])$, where this postcondition is calculated from the forward assignment rule.
  - $wlp(v \coloneqq e,\ p[v_0 \mathbin{/} v] \land v = e[v_0 \mathbin{/} v]) \quad \equiv (p[v_0 \mathbin{/} v] \land v = e[v_0 \mathbin{/} v])[e \mathbin{/} v]$
    $\equiv p[v_0 \mathbin{/} v] \land e = e[v_0 \mathbin{/} v]$
    - $p \land v = v_0 \quad \Leftrightarrow p \land T \land v = v_0$
      $\Leftrightarrow p \land e = e \land v = v_0$
      $\Leftrightarrow p[v \mathbin{/} v] \land e = e[v \mathbin{/} v] \land v = v_0$
      $\Rightarrow p[v_0 \mathbin{/} v] \land e = e[v_0 \mathbin{/} v]$
      $\equiv wlp(v \coloneqq e,\ p[v_0 \mathbin{/} v] \land v = e[v_0 \mathbin{/} v])$

- $(p \wedge v = v_0) \Rightarrow wlp(v \coloneqq e,\ p[v_0 \,/\, v] \wedge v = e[v_0 \,/\, v])$. This implies that $\{p\}\ v \coloneqq e\ \{p[v_0 \,/\, v] \wedge v = e[v_0 \,/\, v]\}$. is a valid triple under partial correctness. Note that, we don't have $(p \wedge v = v_0) \Leftrightarrow wlp(v \coloneqq e,\ p[v_0 \,/\, v] \wedge v = e[v_0 \,/\, v])$.

- Here we showed that these two assignment rules can derive from each other: these two rules are equally strong, if one can create a partial valid triple then the other can also create a partially valid triple. We didn't find anything interesting between sp and wlp in general.